Mesa3D Unveiled: From glDrawArrays(...) to GPU Magic

Christian Gmeiner

2024-09-17 @ OSS-EU

igalia

Who am I?

igalia

Agenda

- Why should I care about Mesa 3D?
- Understanding GPUs
- Let's render a triangle
- Overview of Mesa 3D's architecture
- Shader Compilation in Mesa3D
- Command Stream
- Conclusion

Why should I care about Mesa 3D?

igalia

Why should I care about Mesa 3D?

igalia

Understanding GPUs

igalia

Understanding GPUs

A GPU executes a 'job,' which initializes it to a defined state and processes data through provided 'buffers' during an 'operation'.

Understanding GPUs

The GPU operates in parallel with the CPU, using 'fences' to synchronize tasks between them.

Understanding GPUs

A GPU uses thousands of specialized cores working in parallel to accelerate complex computations.

Understanding GPUs

The role of the kernel driver.

igalia

Understanding GPUs

- /dev/dri/card0
- /dev/dri/card1
- /dev/dri/renderD128

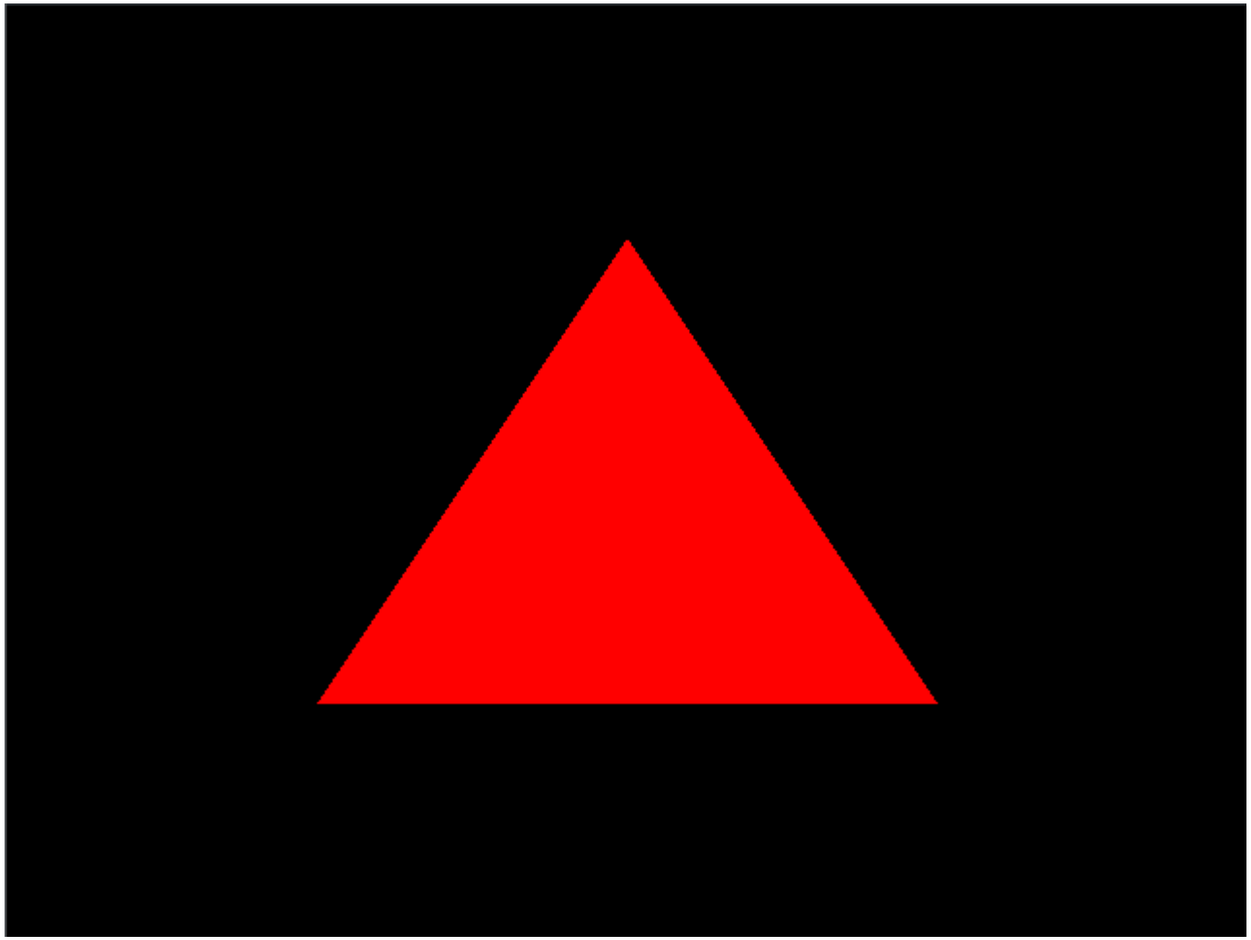
```
[drm] Initialized imx-dcss 1.0.0 20190917 for ... on minc
[drm] Initialized etnaviv 1.4.0 20151214 for etnaviv on m
```


igalia

Understanding GPUs

The whole "job creation" work happens in userspace. This is where Mesa 3D enters the room.

Lets render a triangle



igalia

Lets render a triangle

We need a buffer to provide our vertices for our triangle.

```
float vertices[] = {
    -0.5f, -0.5f, 0.0f,
     0.5f, -0.5f, 0.0f,
     0.0f,  0.5f, 0.0f
};

GLuint VBO;
glGenBuffers(1, &VBO);
glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
```


igalia

Lets render a triangle

Vertex Shader

```
attribute vec3 position;
void main() {
    gl_Position = vec4(position, 1.0);
}
```


igalia

Lets render a triangle

Fragment Shader

```
void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);  // Red colo
}
```


igalia

Lets render a triangle

Create and link shader program

```
GLuint shaderProgram = glCreateProgram();
glAttachShader(shaderProgram, vertexShader);
glAttachShader(shaderProgram, fragmentShader);
glLinkProgram(shaderProgram);
```

Lets render a triangle

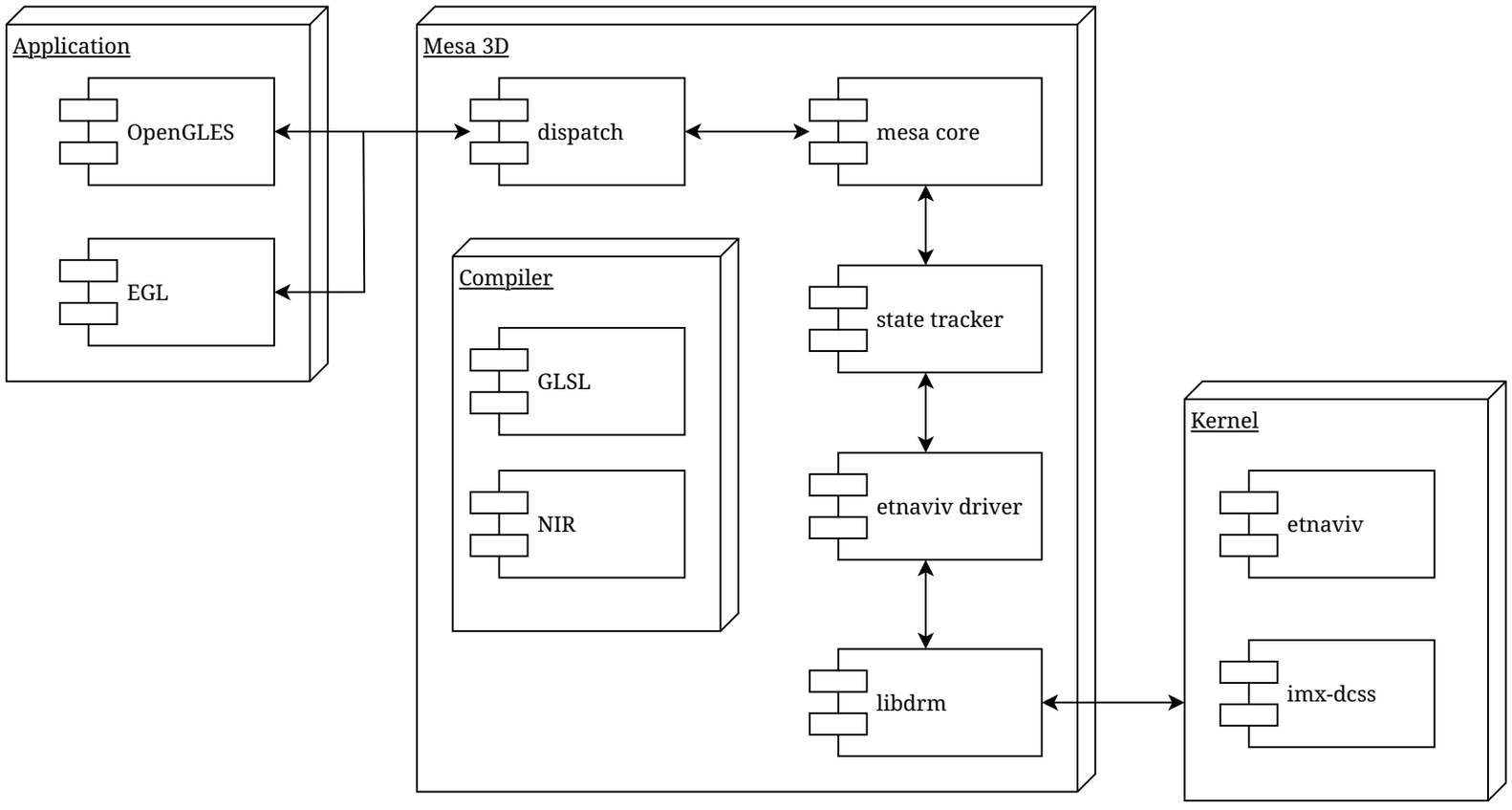
```
glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

glUseProgram(shaderProgram);
glBindBuffer(GL_ARRAY_BUFFER, VBO);
glDrawArrays(GL_TRIANGLES, 0, 3);
```

Overview of Mesa 3D's architecture

igalia

Overview of Mesa 3D's architecture



GL Dispatch

```
glGenBuffers(..);
glDrawArrays(..);
```


igalia

GL Dispatch

Mesa 3D provides a highly optimized dispatch implementation to minimize the overhead.

https://docs.mesa3d.org/dispatch.html

igalia

Mesa Core

```
void GLAPIENTRY
_mesa_GenBuffers(GLsizei n, GLuint *buffers)
{
   GET_CURRENT_CONTEXT(ctx);
   create_buffers_err(ctx, n, buffers, false);
}
```


igalia

State Tracker

Translates the Mesa Core API and functionality into something a bit more flexible and easy to write drivers for.

Gallium

Gallium is essentially an API for writing graphics drivers in a largely device-agnostic fashion.

libdrm

The pice of software that talks to the kernel driver.

Shader Compilation in Mesa3D

We are using some GLSL shaders - what's with them?

Shader Compilation in Mesa3D

GLSL -> NIR -> ASM

igalia

Shader Compilation in Mesa3D

Lets have a look how our GLSL shaders transformed into the final GPU assembly.

igalia

Shader Compilation in Mesa3D

Vertex Shader (GLSL)

```
attribute vec3 position;
void main() {
    gl_Position = vec4(position, 1.0);
}
```


igalia

Shader Compilation in Mesa3D

Vertex Shader (NIR)

```
impl main {
    block b0:  // preds:
    32    %0 = load_const (0x00000000)
    32x3  %1 = @load_input (%0 (0x0)) (...)  // position
    32    %2 = load_const (0x3f800000 = 1.000000)
    32    %3 = deref_var &gl_Position (shader_out vec4)
    32x4  %4 = vec4 %1.x, %1.y, %1.z, %2 (0x3f800000)
               @store_deref (%3, %4) (wrmask=xyzw, access
               // succs: b1
    block b1:
}
```


igalia

Shader Compilation in Mesa3D

Vertex Shader (ASM)

```
000 mov t0.xyz_, void, void, t0.xyzx
001 mov t0.___w, void, void, 1.000000
```


igalia

Shader Compilation in Mesa3D

Fragment Shader (GLSL)

```
void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);  // Red colo
}
```


igalia

Shader Compilation in Mesa3D

Fragment Shader (NIR)

```
impl main {
    block b0:  // preds:
    32x4  %0 = load_const (0x3f800000, 0x00000000, 0x0000
    32    %1 = deref_var &gl_FragColor (shader_out vec4)
               @store_deref (%1, %0 (1.000000, 0.000000,
               // succs: b1
    block b1:
}
```

Shader Compilation in Mesa3D

Fragment Shader (ASM)

```
000 mov t1, void, void, u0.xyyx
```


igalia

Command Stream

The "job" description.

igalia

Command Stream

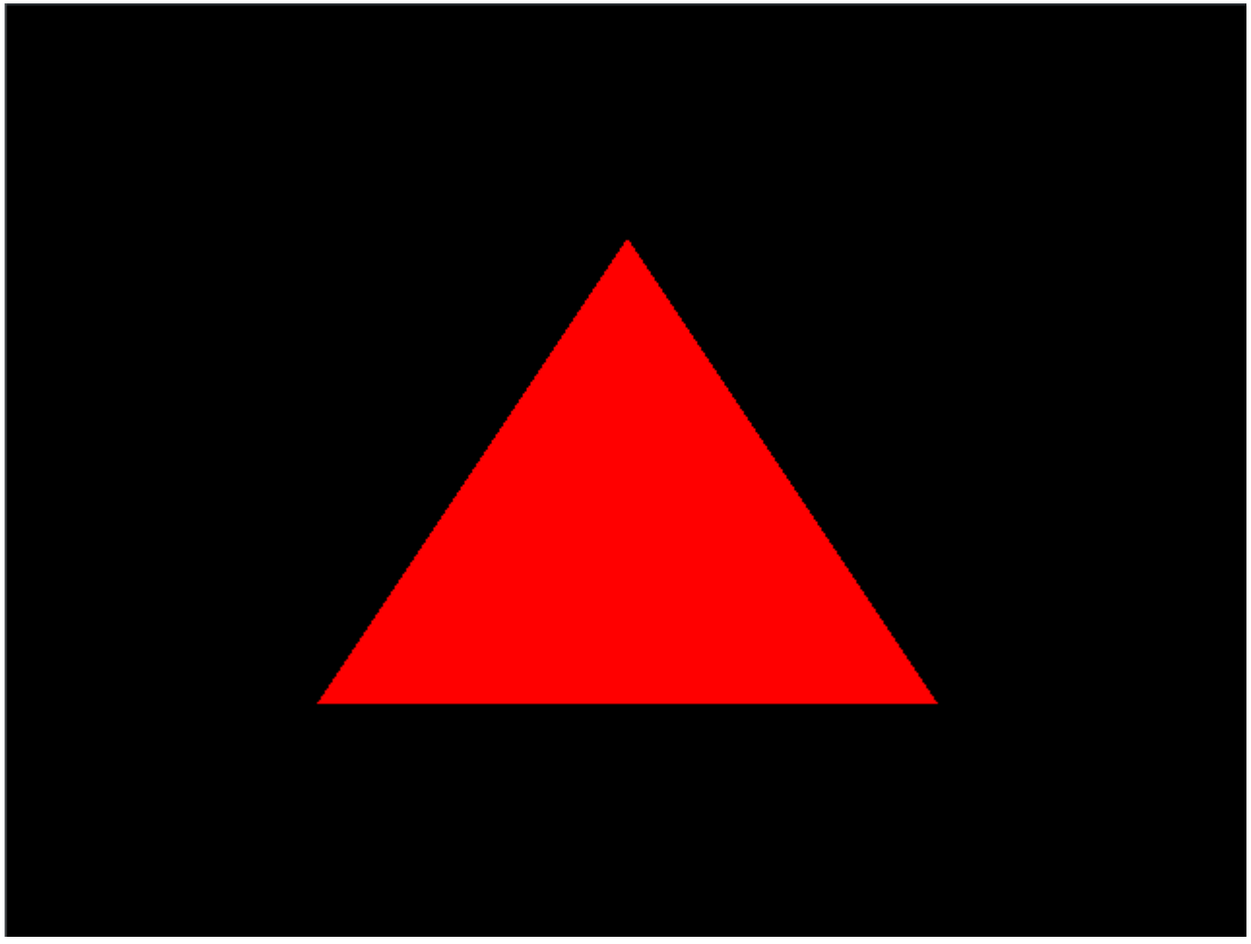
- GPU states are set
- buffer objects are referenced
- one or more draw operations are defined

Command Stream

```
...
0xffc2e000, /*   PE.PIPE[0].COLOR_ADDR := *0xffc2e000 */
0xffc2e000, /*   PE.PIPE[1].COLOR_ADDR := *0xffc2e000 */
...
0xffd70000, /*   NFE.VERTEX_STREAMS[0].BASE_ADDR := *0xf
...
0xffd6f000, /*   VS.INST_ADDR := *0xffd6f000 */
...
0xffd6e000, /*   PS.INST_ADDR := *0xffd6e000 */
...
0x60040001, /* DRAW_INSTANCED (12) ... TYPE=TRIANGLES,INS
0x00000003, /*   COUNT INSTANCE_COUNT_HI=0x0,VERTEX_COUNT
0x00000000, /*   START INDEX=0x0 */
```


igalia

Conclusion



igalia

Thanks

igalia

Q&A

igalia

igalia